FORTH

INSTITUTE OF COMPUTER SCIENCE

Demystifying the RISC-V Linux software stack

Nick Kossifidis - Spring 2022 RISC-V Week, Paris

Adding a new arch on Linux...

Linux is huge !

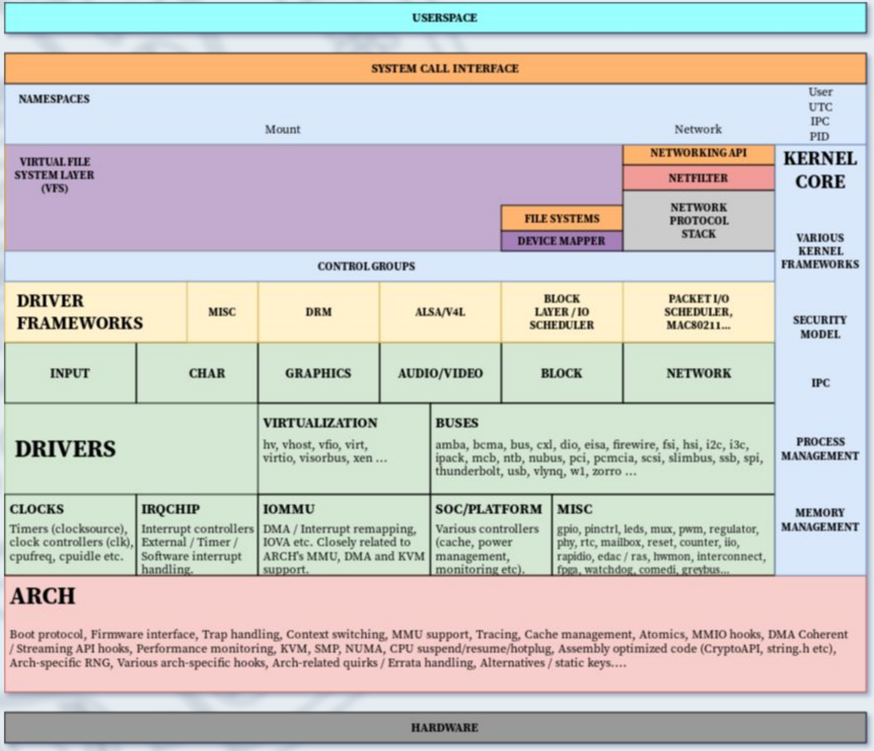
~23 million lines of code and counting

~64k files in the kernel tree

Almost 2.000 developers / release

More than 20.000 developers have contributed so far

A visualisation attempt



USERSPACE

SYSTEM CALL INTERFACE

NAMESPACES

Mount

Network

User
UTC
IPC
PID

VIRTUAL FILE SYSTEM LAYER (VFS)

NETWORKING API

NETFILTER

NETWORK PROTOCOL STACK

FILE SYSTEMS

DEVICE MAPPER

KERNEL CORE

VARIOUS KERNEL FRAMEWORKS

SECURITY MODEL

IPC

PROCESS MANAGEMENT

MEMORY MANAGEMENT

CONTROL GROUPS

DRIVER FRAMEWORKS

MISC

DRM

ALSA/V4L

BLOCK LAYER / IO SCHEDULER

PACKET I/O SCHEDULER, MAC80211...

INPUT

CHAR

GRAPHICS

AUDIO/VIDEO

BLOCK

NETWORK

DRIVERS

VIRTUALIZATION

hv, vhost, vfio, virt, virtio, visorbus, xen ...

BUSES

amba, bcma, bus, cxl, dio, eisa, firewire, fsi, hsi, i2c, i3c, ipack, mcb, ntb, nubus, pci, pcmcia, scsi, slimbus, ssb, spi, thunderbolt, usb, vlynq, w1, zorro ...

CLOCKS

Timers (clocksource), clock controllers (clk), cpufreq, cpuidle etc.

IRQCHIP

Interrupt controllers External / Timer / Software interrupt handling.

IOMMU

DMA / Interrupt remapping, IOVA etc. Closely related to ARCH's MMU, DMA and KVM support.

SOC/PLATFORM

Various controllers (cache, power management, monitoring etc).

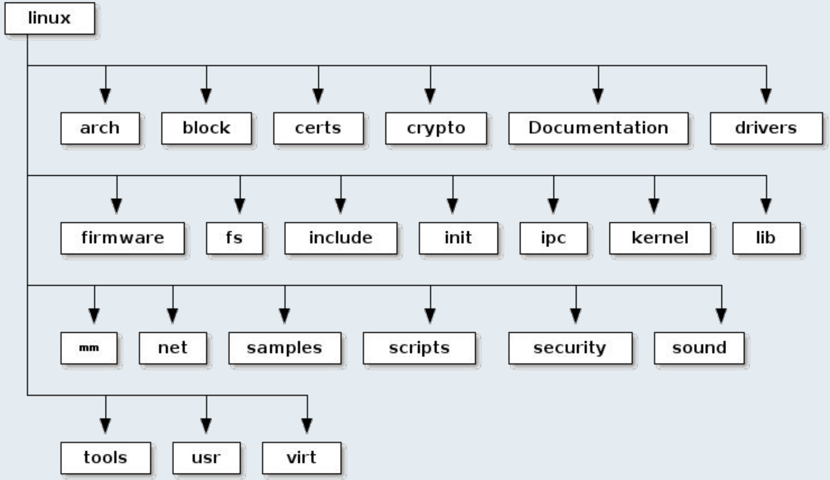
MISC

gpio, pinctrl, leds, mux, pwm, regulator, phy, rtc, mailbox, reset, counter, iio, rapidio, edac / ras, hwmon, interconnect, fpga, watchdog, comedi, greybus...

ARCH

Boot protocol, Firmware interface, Trap handling, Context switching, MMU support, Tracing, Cache management, Atomics, MMIO hooks, DMA Coherent / Streaming API hooks, Performance monitoring, KVM, SMP, NUMA, CPU suspend/resume/hotplug, Assembly optimized code (CryptoAPI, string.h etc), Arch-specific RNG, Various arch-specific hooks, Arch-related quirks / Errata handling, Alternatives / static keys....

HARDWARE



linux

arch
block
certs
crypto
Documentation
drivers

firmware
fs
include
init
ipc
kernel
lib

mm
net
samples
scripts
security
sound

tools
usr
virt

Comparing /arch to the rest of the kernel

/arch -> ~1.7mloc

/{kernel, lib, mm, ipc, init} -> ~0.5mloc

/{block, net, crypto, security, fs} -> ~2mloc

/{drivers, sound} -> ~16mloc

/arch/x86 -> ~290kloc

/arch/arm -> ~272kloc

/arch/arm64 -> ~85kloc

/arch/riscv -> ~23kloc

Under the hood

RISC-V Privilege modes

Machine Mode

- Mandatory
- The most privileged / protected mode visible to the software (there is also Debug mode but it's only accessible / visible to hw debuggers)
- Physical memory addressing
- Physical memory protection
- Trap/Interrupt handling and delegation

User Mode

- Optional (depends on M-mode)
- The least privileged / protected mode
- Physical/virtual memory addressing Physical/virtual memory protection
- No trap/interrupt handling

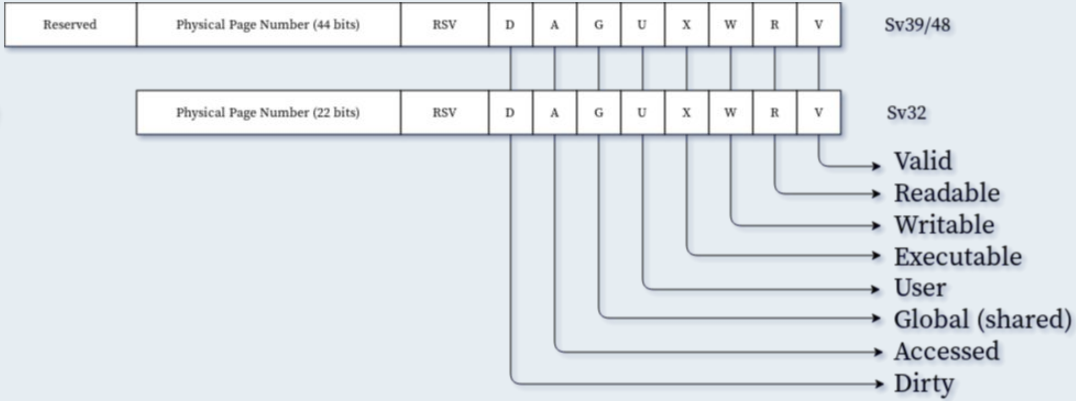
Supervisor Mode

- Optional (depends on M-mode and U-mode)
- Sits between M-mode and U-mode
- Provides virtual memory addressing / protection
- Trap/interrupt handling through delegation, managed by M-mode
- May act as a hypervisor (aka HS-mode) through the use of an extra set of CSRs, also providing a second stage of translation / protection for guests (aka VS-mode instances)

The RISC-V Privileged Spec
https://github.com/riscv/riscv-isa-manual/releases

RISC-V Virtual memory

- 3 level page table for RV32 (Sv32)
- 3, 4, 5 level page tables for RV64 (Sv39/48/57)
- 2nd stage of translation for VS mode (G-stage), managed by HS mode
- NAPOT encoding available
- Up to 16bit ASID
- SMEP always active
- SMAP controlled by sstatus.SUM bit
- Page-based memory types (Non Cacheable, I/O) for RV64



Reserved	Physical Page Number (44 bits)	RSV	D	A	G	U	X	W	R	V	Sv39/48
	Physical Page Number (22 bits)	RSV	D	A	G	U	X	W	R	V	Sv32

Valid
Readable
Writable
Executable
User
Global (shared)
Accessed
Dirty

Facilities on M-mode

Provide infos on the current hardware thread (hart):

- Vendor id (mvendorid), Microarchitecture id (marchid), Implementation id (mimpid)
- Current hart’s id (hartid)
- Available hart extensions (misa, menvcfg)
- Pointer to the configuration structure (mconfigptr) from which we can also generate the device tree or ACPI tables

Configure hart extensions (misa, menvcfg, mstatus) and security features (mseccfg)

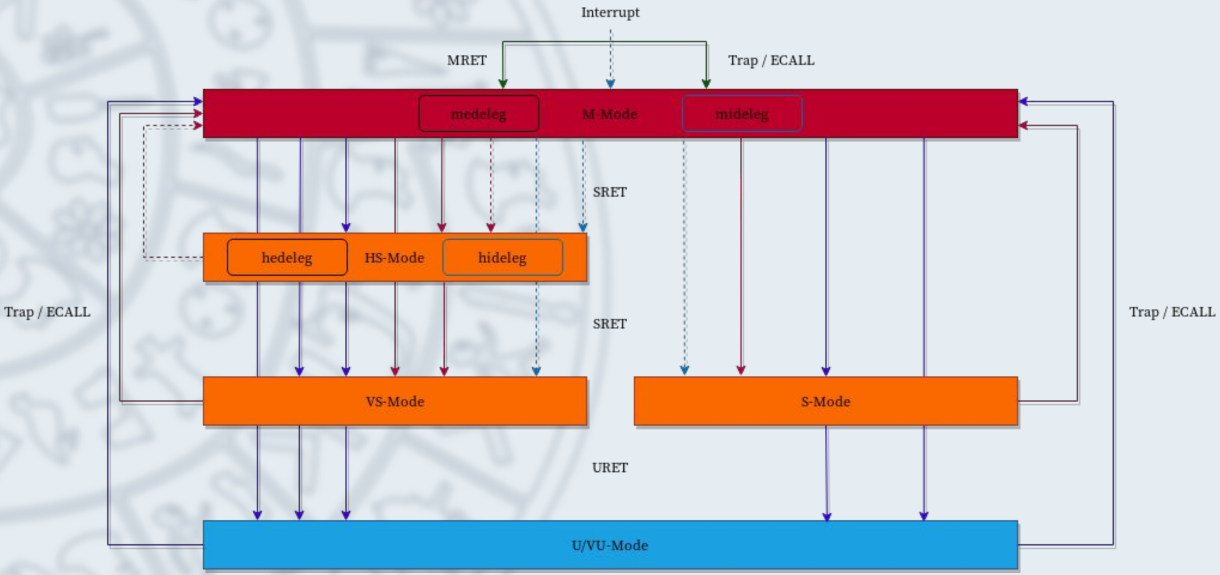
Physical Memory Protection (PMP/ePMP)

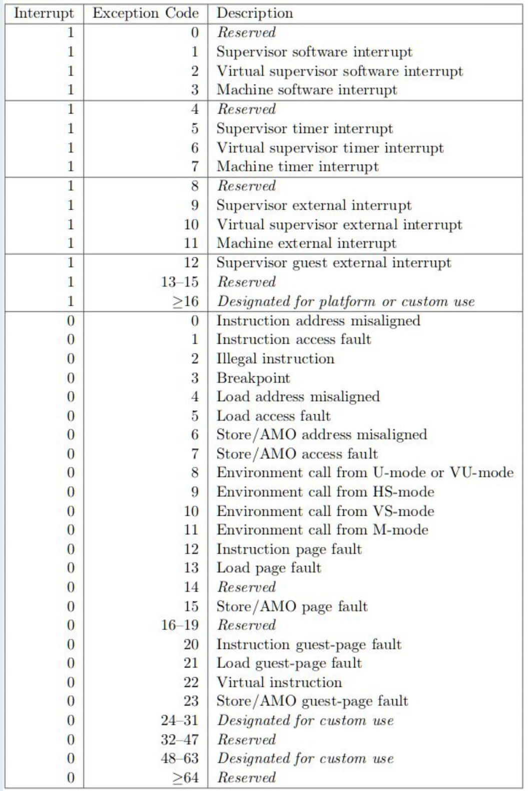
Configure profile counters

Fixed-frequency timer (mtime) with the ability to schedule timer interrupts (mtimecmp)

Configure trap and interrupt auto-delegation to S / HS modes

RISC-V Trap and interrupt delegation / mode switching





Interrupt	Exception Code	Description
1	0	*Reserved*
1	1	Supervisor software interrupt
1	2	Virtual supervisor software interrupt
1	3	Machine software interrupt
1	4	*Reserved*
1	5	Supervisor timer interrupt
1	6	Virtual supervisor timer interrupt
1	7	Machine timer interrupt
1	8	*Reserved*
1	9	Supervisor external interrupt
1	10	Virtual supervisor external interrupt
1	11	Machine external interrupt
1	12	Supervisor guest external interrupt
1	13–15	*Reserved*
1	≥16	*Designated for platform or custom use*
0	0	Instruction address misaligned
0	1	Instruction access fault
0	2	Illegal instruction
0	3	Breakpoint
0	4	Load address misaligned
0	5	Load access fault
0	6	Store/AMO address misaligned
0	7	Store/AMO access fault
0	8	Environment call from U-mode or VU-mode
0	9	Environment call from HS-mode
0	10	Environment call from VS-mode
0	11	Environment call from M-mode
0	12	Instruction page fault
0	13	Load page fault
0	14	*Reserved*
0	15	Store/AMO page fault
0	16–19	*Reserved*
0	20	Instruction guest-page fault
0	21	Load guest-page fault
0	22	Virtual instruction
0	23	Store/AMO guest-page fault
0	24–31	*Designated for custom use*
0	32–47	*Reserved*
0	48–63	*Designated for custom use*
0	≥64	*Reserved*

RISC-V Interrupt delivery

The old way (SiFive CLINT / PLIC)

- Wired interrupts only, no MSIs
- Shared between privilege modes
- Directly to M-mode and then delegated (so even S-mode software interrupts go through M-mode)
- No virtualization support

The new way (RISC-V ACLINT / AIA)

- Both wired and MSIs
- Different interrupt settings per privilege mode
- Interrupts delivered to specific privilege modes
- Virtualization support

For more information:
https://github.com/riscv/riscv-aia
https://github.com/riscv/riscv-aclint
Advanced Interrupt Architecture and Advanced CLINT
Anup Patel, John Hauser - RISC-V Summit 2021
(https://www.youtube.com/watch?v=je9Qr23mclU)

Firmware architecture

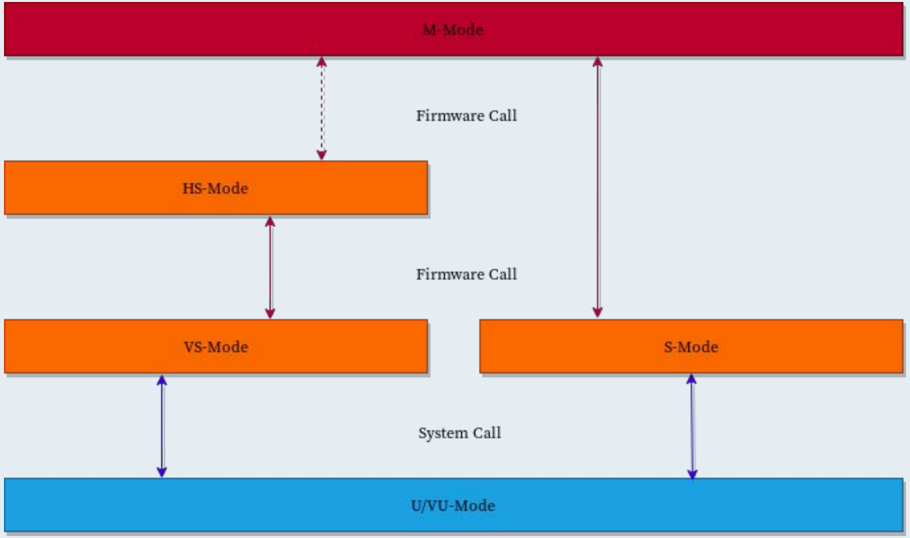
Supervisor Binary Interface (SBI)

Firmware call API

- S-Mode <-> M-Mode
- HS-Mode <-> M-Mode
- VS-Mode <-> HS-Mode

Available services:

- Provide access to M-mode facilities
  - Timer, PMU, hart/imp/vendor IDs…
- Inter-Processor Interrupts (IPI)
- Remote Fence (memory barrier)
- Hart State Management (suspend/resume)
- System Reset
- …



For more information: https://github.com/riscv-non-isa/riscv-sbi-doc

INSTITUTE OF COMPUTER SCIENCE

RISC-V OS Boot protocol

Direct:

- Get Device Tree through a1 gp register
- Get Hart ID through a0 gp register

RISC-V Device Tree bindings under Documentation/bindings:
/riscv/cpus.yaml
/interrupt-controller/riscv,cpu-intc.txt
/interrupt-controller/sifive,plic-1.0.0.yaml
/cpu/cpu-topology.txt

For more information:
Atish Pattra - An introduction to RISC-V Boot flow (RISC-V Summit 2019)
https://www.youtube.com/watch?v=sPjtvqfGjnY
https://archive.fosdem.org/2021/schedule/event/firmware_uor/
https://github.com/riscv-admin/riscv-uefi-edk2-docs
arch/riscv/kernel/head.S
drivers/firmware/efi/libstub/riscv-stub.c

EFI stub:

- Get Device Tree through EFI Config Table
- Get Hart ID through the device tree's chosen/boot-hartid or through the new RISCV_EFI_BOOT_PROTOCOL

Runtime firmware implementations

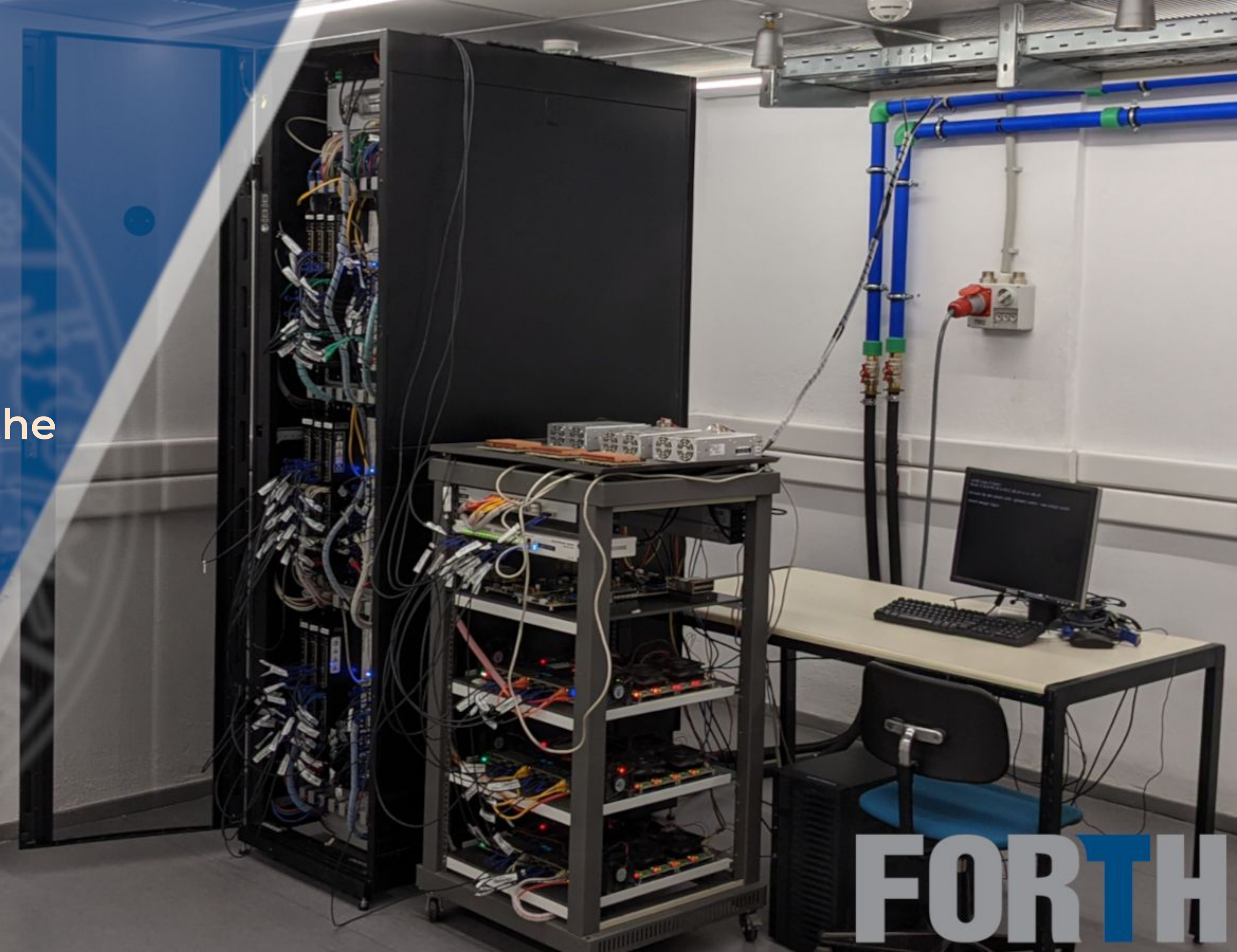
Reference implementation: OpenSBI
Can act as a standalone firmware / first stage boot loader
Can be used as a library for other runtime firmware implementations
    Used on EDK2 (EFI Runtime firmware)
Can be used for static partitioning of the system (OpenSBI Domains)

Other implementations of the SBI spec

- Hypervisors (to provide SBI for their guests):
  - KVM, Xvisor, Diosix
- RustSBI
- Coffer (Secure monitor)

Useful links:
https://github.com/riscv-software-src/opensbi
https://github.com/xvisor/xvisor
https://diosix.org/
https://github.com/rustsbi/rustsbi
https://github.com/jwnhy/coffer

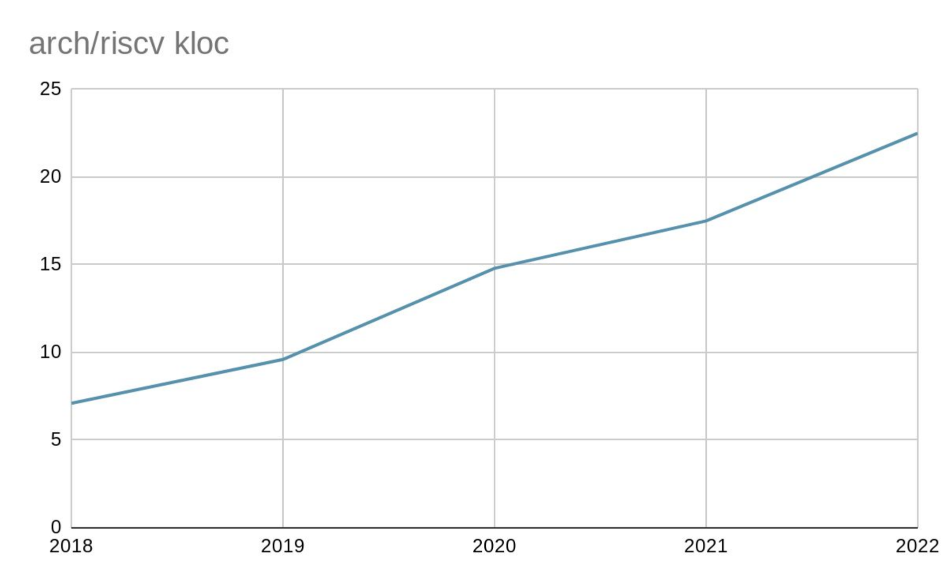
Current status of the RISC-V Linux port



The RISC-V Linux port

- Git repository:
  - https://git.kernel.org/pub/scm/linux/kernel/git/riscv/linux.git
- Documentation:
  - https://docs.kernel.org/riscv/index.html
- Use a recent gcc toolchain:
  - E.g.: https://github.com/riscv-collab/riscv-gnu-toolchain
- Supported systems on mainline:
  - QEMU RISC-V Virt machine
  - SiFive Hifive Unleashed / Unmatched
  - PolarFire SoC FPGA Icicle Kit
  - Kendryte 210 (NOMMU, Linux runs on M-mode)
  - Allwinner D1 (Work in Progress)
- Supported systems with custom patches
  - LowRISC
  - Beagle-V
  - …

The evolution of /arch/riscv



RISC-V Extension support

Virtual memory extensions:

- Sv32, Sv39 / Sv32x4, Sv39x4 supported
- Sv48 / Sv48x4 supported (Alex Ghiti, Anup Patel)
- Sv57 / Sv57x4 supported (Qinglin Pan, Anup Patel)
- ASID allocator supported (Anup Patel)
- Page-based memory types (Svpbmt) submitted (v9 Heiko Stuebner)
- Fast TLB invalidation (Svinval) submitted (v2 Mayuresh Chitale)
- NAPOT pages (Svnapot) submitted (v1 Qinglin Pan)

H(ypervisor) extension supported (Anup Patel)

V(ector) extension submitted (v9 Greentime Hu, Guo Ren, Vincent Chen)

S and VS level Time Compare (Sstc) submitted (v3 Atish Patra, also v2 on OpenSBI)

Cache Management Operations Base (Zicbo*) submitted (v1 Heiko Stuebner -waiting for toolchain support)

Profiling through PMU (including Sscofpmf) supported (Atish Patra)

Pending:

- Scalar crypto support on CryptoAPI
- Arch-specific RNG (Zkr from scalar crypto)

Linux features support

Feature status (30/42): https://docs.kernel.org/riscv/features.html

Work in progress:

- Queued RWlocks submitted (v4 Palmer Dabbelt)
- HAVE_ARCH_HUGE_VMAP (v2 Liu Shixin, also part of Svnapot patchset)

Supported features not included on features list:

- CPU Idle, CPU Hot Plugging (through SBI HSM) (Anup Patel, Atish Patra)
- kexec/kdump (Nick Kossifidis)
- Alternatives framework (Vincent Chen, Heiko Stuebner)
- Restartable Sequences (RSEQ) (Vincent Chen)
- Strict kernel/module RWX (Zong Li, Jisheng Zhang)
- XIP image (Alex Ghiti)
- NOMMU support (userspace is WiP) (Christoph Hellwig)
- NUMA (Greentime Hu)
- EFI Stub (Atish Patra)
- …

The development process…

It’s a challenge for everyone involved

RISC-V is a unique architecture

We are not one company that has control over the whole process

Everything happens in parallel, the spec, the simulators, the toolchain etc

We have limited hardware to play with, development happens mostly on QEMU

Due to the modular nature of RISC-V we need to support all possible configurations

We need to support vendor extensions on top of standard extensions

We need to handle hw errata in sw

We need to be able to provide one image that boots everywhere

But things are getting better

Check out the talks from Philipp Tomsich and Mark Himelstein on Thursday

A few examples to get an idea

Cache management operations & Page-based memory types

Vector support

Kendryte 210 draft MMU

Allwinner D1 noncompliant Sv39

A few examples from EPAC to get an idea

Unique memory layout

Device tree outside of linear mapping

1:1 physical/virtual mapping (used for Sv48/Sv57 detection) is invalid in our case

Kdump worked fine on QEMU but not on HiFive boards

For more information on EPAC check out tomorrow’s talk from Jesús Labarta

The Accelerator Tile of European Processor Initiative

Alternatives and static keys

Alternatives framework:

Provide alternative assembly code snippets that can override the default implementation through in-memory live patching of the kernel image.

Example:

Implement standard page-based memory types on supported implementations

Implement custom Allwinner D1 equivalent

Static keys:

Same concept of live patching but focused on branches

The heroes…

Top 10 contributors:

1.	97	Atish Patra
2.	91	Palmer Dabbelt
3.	87	Christoph Hellwig
4.	84	Anup Patel
5.	60	Jisheng Zhang
6.	59	Kefeng Wang
7.	55	Zong Li
8.	54	Alexandre Ghiti
9.	37	Vincent Chen
10.	30	Damien Le Moal

Next steps...

FORTH

Upcoming features

IOMMU

Check out RISC-V IOMMU Architecture Overview from Perrine Peresse on Thursday

Secure Boot / Root of trust

Vector crypto

AP-TEE architecture

MPU

IOPMP

Check out RISC-V : Securing the Future of Open Source Computing from Andrew Dellow on Thursday

And many more

Check out State of the Union & the Road Ahead from Mark Himelstein on Thursday

Questions ?

Thank you !

FORTH